Plates4U: A Collaborative-Filtering approach to Recipe Recommenders

Anthony Fong
UC San Diego
San Diego, California
ajf010@ucsd.edu

Alex Pham
UC San Diego
San Diego, California
alp075@ucsd.edu

Zachary Nguyen
UC San Diego
La Jolla, California
zanguyen@ucsd.edu

Abstract

Existing options for recipe recommendations are less than satisfactory. We sought to solve this problem by creating our own recommendation system hosted on a website. Using recipe data from Food.com, we created a classifier to identify cuisines of recipes, a popularity based recommender, and a content-based filtering recommender using cosine similarity. In the future, we would like to improve upon this recommender by exploring alternative ways to model ingredients, by tracking implicit/explicit data of a user, and by creating a hybrid recommender using collaborative techniques.

1 Introduction

A recommender system is an important application of data science that serves to generate item recommendations that would be interesting and relevant to a user. Recommender systems are integral to the successes of the music, news, and even social networking industries, but its applications in the recipe/culinary industry still have room for improvement.

With our hectic and unpredictable daily schedules, it can be easy to rely on convenient and unhealthy foods that detriment our health. And oftentimes, indecisiveness in meal preparation can lead to monotonous and unenjoyable meals. Typically, in searching for a recipe, one either finds very tedious recipe blogs that take too much time to actually view the recipe details, or they find overly complicated recipe websites that can require a membership or profile in order to access recipes. Finding great recipes that accommodate your preferences should not be this difficult, and so the goal of this project was to develop a recipe recommender system that suggests interesting recipes based on filters that the user could input such as ingredients, cook time, and cuisine type. The final output of our project is a simple website where users can easily select filters to generate recipes.

2 Target Goal

The target of our recommender system using the recipe datasets we have gathered, is to be able to give recipe recommendations given limited information about the user. The goal is to make the recommender as user friendly as possible removing the need to create any accounts or feel any sort of long-term commitment to the model.

The recommender should recommend recipes to users given ingredients the user has lying around or plans to get sometime soon. This model prevents food waste by giving suggestions as to what users could do with leftover ingredients. It also diversifies the user's palate by giving recipe suggestions that may not have been known to the user beforehand.

If we are able to provide recipe recommendations that both allow users to utilize the ingredients they have lying around, and try new recipes that they may not have had before then our recommender system would have done its job.

3 Dataset

The main dataset we are using for our project was acquired from a Kaggle user named Shuyang Li [1], but the dataset was scraped from Food.com. The dataset contains 8 files. Three of the files from the dataset were pre-processed files that the Kaggle user created for their specific project. Another three files contained the train, test, and validation sets the Kaggle user used in training and testing their models. We did not use any of these 6 files in our project. Instead, we used the remaining two files. The first file contained the raw recipes scraped from Food.com. This file was the scraped recipe data which

held a variety of information about each recipe from Food.com. There is information on the amount of time each recipe takes, what the nutritional information about the recipe is, what users have tagged the recipe with, etc. The second file contained interaction data. This file had information about what user tried what recipe and what they rated it. There is also information as to when the review was submitted.

We also utilized a second dataset from Kaggle uploaded by Kaggle [2]. This second dataset was used to fill in some of the shortcomings of the first dataset. The first dataset had a variety of information about each recipe, but lacked a column specifying what type of cuisine it fell under. This second dataset from Kaggle had data on different recipes, what ingredients are used, and what cuisine type they fell under. Using the ingredients data from the second dataset, we were able to train a Random Forest Classifier up to ~75% accuracy to predict cuisine type of each recipe in the first dataset.

This is not a perfect solution and carries with it its own set of limitations. Since not every recipe in the first dataset appeared in the second dataset, we could not directly map cuisine types from the second dataset to the first. What this means is that we could be mislabeling many of the recipes in the first dataset without any way of verifying. There are also cases where two recipes seem very similar and are likely the same, but have different names. The different names could be a factor of which culture or person named the recipe, and as such is difficult to identify. This also prevented us from directly mapping recipe cuisine types from the second dataset to the first. Though even with the two issues we faced the model still performed relatively well on the test data provided, and so the model should be generalizable. Some example recipes and their cuisine classifications are shown below:

recipe_id	name	minutes	nutrition	n_steps	steps	description	ingredients	n_ingredients	mean_rating	ingredient_list	cuisine
33606	italian sandwich pasta salad	25	[279.2, 14.0, 12.0, 22.0, 23.0, 11.0, 12.0]	6	['cook pasta and set aside', 'place onions , d...	this is a fun salad that uses the same ingredi...	['tri-color spiral pasta', 'dill pickles', 'ri...	16	5.00	['tri-color spiral pasta', 'dill pickles', 'ri...	italian
94710	italian fries	15	[241.0, 12.0, 1.0, 23.0, 10.0, 11.0, 12.0]	4	['deep fry french fries according to package d...	ok, so this is not "really" italian, but it is...	['frozen french fries', 'oil', 'salt & freshly...	7	5.00	['frozen french fries', 'oil', 'salt & freshly...	italian
35173	italian gut busters	45	[816.9, 100.0, 11.0, 97.0, 98.0, 180.0, 2.0]	7	['lay out sandwich rolls on jelly roll pans / ...	my sister-in-law made these for us at a family...	['sandwich bun', 'good seasonings italian sala...	9	4.00	['sandwich bun', 'good seasonings italian sala...	italian
83025	jeanne s style birthday cake	230	[5467.4, 516.0, 1196.0, 135.0, 110.0, 615.0, 1...	25	['to prepare base , cut shortening into dry in...	a bakery in winnipeg is famous for this specia...	['shortening', 'icing sugar', 'vanilla', 'all-...	10	2.29	['shortening', 'icing sugar', 'vanilla', 'all-...	southern_us
52804	jiffy extra moist carrot cake	50	[612.1, 49.0, 170.0, 25.0, 15.0, 39.0, 25.0]	8	['preheat oven to 350 degrees', 'mix together ...	this is a very tasty, moist, carrot cake, a ni...	['yellow cake mix', 'vanilla instant pudding m...	11	4.00	['yellow cake mix', 'vanilla instant pudding m...	southern_us

Image 1: Sample dataset

As you can see the recipes are classified as expected, with both "italian sandwich pasta salad" and "italian fries" being classified as italian cuisines.

Since both datasets were found on Kaggle, the data was already close to being completely clean and did not require any pre-processing. The only thing we did to clean the data was we removed some recipes from the dataset that had ridiculously long cooking times of several months or years. We assumed that these recipes were posted by trolls, and were not actually real recipes. This is an assumption we made as a group, however, as it is possible that these are real recipes that we are just not familiar with. Though even if we did not have to clean the data much, we still conducted some simple exploratory data analysis to gain a better understanding of the datasets.

	name	id	minutes	contributor_id	submitted	tags	nutrition	n_steps	steps	description	ingredients	n_ingredients
0	arriba baked winter squash mexican style	137739	55	47892	2005-09-16	['60-minutes-or-less', 'time-to-make', 'course...	[51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]	11	['make a choice and proceed with recipe', 'dep...	autumn is my favorite time of year to cook! th...	['winter squash', 'mexican seasoning', 'mixed ...	7
1	a bit different breakfast pizza	31490	30	26278	2002-06-17	['30-minutes-or-less', 'time-to-make', 'course...	[173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]	9	['preheat oven to 425 degrees f', 'press dough...	this recipe calls for the crust to be prebaked...	['prepared pizza crust', 'sausage patty', 'egg...	6
2	all in the kitchen chili	112140	130	196586	2005-02-25	['time-to-make', 'course', 'preparation', 'mai...	[269.8, 22.0, 32.0, 48.0, 39.0, 27.0, 5.0]	6	['brown ground beef in large pot', 'add choppe...	this modified version of 'mom's' chili was a h...	['ground beef', 'yellow onions', 'diced tomato...	13
3	alouette potatoes	59389	45	68585	2003-04-14	['60-minutes-or-less', 'time-to-make', 'course...	[368.1, 17.0, 10.0, 2.0, 14.0, 8.0, 20.0]	11	['place potatoes in a large pot of lightly sal...	this is a super easy, great tasting, make ahea...	['spreadable cheese with garlic and herbs', 'n...	11
4	amish tomato ketchup for canning	44061	190	41706	2002-10-25	['weeknight', 'time-to-make', 'course', 'main-...	[352.9, 1.0, 337.0, 23.0, 3.0, 0.0, 28.0]	5	['mix all ingredients& boil for 2 1 / 2 hours ...	my dh's amish mother raised him on this recipe...	['tomato juice', 'apple cider vinegar', 'sugar...	8

Figure 1: The format of the recipe dataset

Figure 1 depicts a quick snapshot of what the recipe dataset looks like. There are 12 different columns with 236,637 rows. Each column title very accurately describes what is stored in that column.

	user_id	recipe_id	date	rating	review
0	38094	40893	2003-02-17	4	Great with a salad. Cooked on top of stove for...
1	1293707	40893	2011-12-21	5	So simple, so delicious! Great for chilly fall...
2	8937	44394	2002-12-01	4	This worked very well and is EASY. I used not...
3	126440	85009	2010-02-27	5	I made the Mexican topping and took it to bunk...
4	57222	85009	2011-10-01	5	Made the cheddar bacon topping, adding a sprin...
...	...	...	...	...	...
1132362	116593	72730	2003-12-09	0	Another approach is to start making sauce with...
1132363	583662	386618	2009-09-29	5	These were so delicious! My husband and I tru...
1132364	157126	78003	2008-06-23	5	WOW! Sometimes I don't take the time to rate ...
1132365	53932	78003	2009-01-11	4	Very good! I used regular port as well. The ...
1132366	2001868099	78003	2017-12-18	5	I am so glad I googled and found this here. Th...

1132367 rows × 5 columns

Figure 2: The format of the user interaction dataset

Figure 2 depicts a quick snapshot of what the user interaction dataset looks like. There are 5 different columns with over 1 million rows. Each column title for this dataset also accurately describes what is stored in that column.

	minutes	n_steps	n_ingredients
count	231635.000000	231635.000000	231635.000000
mean	123.108144	9.765506	9.051188
std	1977.767905	5.995153	3.734782
min	0.000000	0.000000	1.000000
25%	20.000000	6.000000	6.000000
50%	40.000000	9.000000	9.000000
75%	65.000000	12.000000	11.000000
max	288000.000000	145.000000	43.000000

Figure 3: A table displaying the different descriptive statistics from the recipe dataset

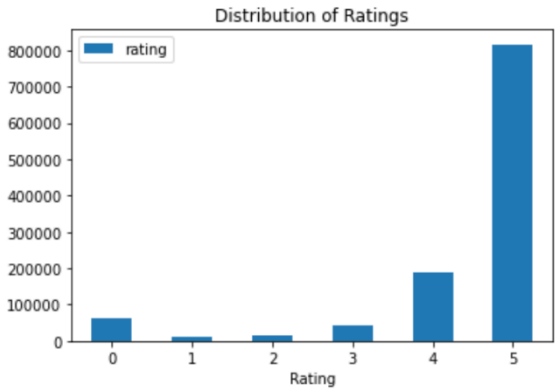

Figure 4: A bar chart with the distribution of ratings

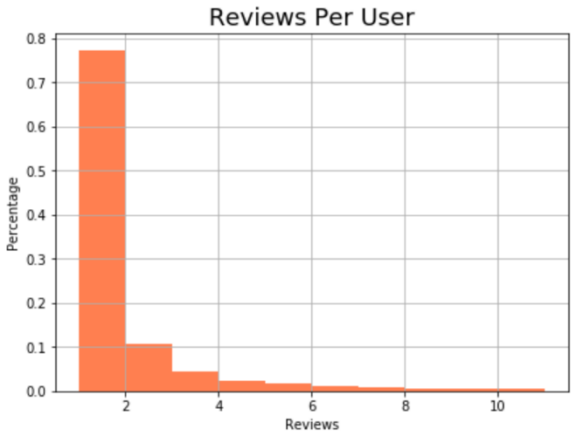

Figure 5: A histogram depicting the number of reviews left by most users

From figures 4 and 5 it is clear that the data is somewhat skewed. The distribution of ratings appears to be left skewed while the number of reviews by users appears to be right skewed. The data does not appear to be normally distributed and most users appear only a few times. With so little individual data it is likely going to be very difficult to build a strong recommender model based on these user interactions. Furthermore the skewed ratings could cause a bias in the recommender model where most recommendations are predicted to have high ratings.

4 Method

For this project the goal is to provide viable recipe recommendations to users of our models. To do this we have set a two stage model setup. We have a most popular model running as the secondary model and cosine similarity running as the main model. The cosine similarity model provides the bulk of the recommendations, but when there are edge cases or when the user does not input enough information to return good recommendations the most popular model substitutes to generate the remaining recommendations.

5 Model

Two models were developed to offer recipe recommendations to users. The first model is a simple top popular recommender that predicts the most popular recipe to all users, and the second model which acts as

the main model utilizes Cosine Similarity to conduct a form of Collaborative Filtering.

5.1 Most Popular (MP)

This trivial baseline model is a most popular prediction model. This model worked by calculating the average rating of every recipe type normalized by the number of users that have tried the recipe. This would give every recipe a rank of sorts. Once this rank is decided the model would predict whether a user would like a recipe based on how it ranked overall. The model only recommends high ranking recipes to users, and as such it is not personalized. Though surprisingly the model still performs relatively well. Using ratings of 4 stars and higher as the user liked it, and 3 stars and lower as the user does not, the most popular recommender still scored a relatively low balanced error rate of 0.34. This makes sense, however, as the recipes are the most popular for a reason so it should not be surprising that many users would like them.

This model was chosen as it is simple to implement, and pretty accurate given its simplicity. The model is also scalable and has no parameters to tune making it an ideal choice for a baseline model or a model used in an AdaBoost algorithm. An AdaBoost algorithm is an algorithm that compiles multiple quick and easy to implement machine learning models with subpar results into a combined model with much better results.

5.2 Cosine Similarity

To deliver a set of recommendations, we use content-based filtering as our method of candidate generation. A content-based filtering method compares items to other items based on their features to calculate a similarity score. Using this similarity score in addition to explicit information a user has given, we can return a set of recommendations. For our implementation, we used cosine similarity as our measure.

$$\text{cosine similarity(A,B)} = \frac{\sum_{i}^{n} A_i \times B_i}{\sqrt{\sum_{i}^{n} A_i^{\,2}} \times \sqrt{\sum_{i}^{n} B_i^{\,2}}} \quad (2)$$

Where:

n: number of ingredients

A_i: the i[th] value of ingredient vector A

B_i: the i[th] value of ingredient vector B

Our cosine similarity formula relies on the input of two ingredient vectors A and B. An ingredient vector is an embedding of a list of ingredients for a given recipe. A user interacts with our recommender by inputting a list of ingredients that are available in their kitchen. This a text input that is comma separated. We then take this input and transform it into a list of strings. This list of strings is then transformed using a MultiLabelBinarizer that has been trained on an entire dataset of recipes that each have a list of ingredients. The transformed input is now a vector of n dimensions, n being the total number of unique ingredients in the dataset. The value of each element is binary. This transformation is very similar to a bag of words model, as we only care about the presence of an ingredient in a recipe.

This process occurs for both the input ingredients and every recipe in the dataset. This places every recipe as a vector in an n dimensional space. Cosine similarity is then used to help us determine a distance between vectors. The vector nearest to the user's input vector is considered the most similar to it.

Once the cosine similarity has been calculated for every recipe, filtering is done to exclude recipes based on the user's input. For example, a user may decide they want only italian dishes, so any recipes that are not italian are removed. This filtering is done for cuisine and cook time.

Finally, the remaining recipes are then sorted in descending order based on similarity and we output the top five recommendations.

5.3 Model Comparison

The most popular model is the easiest model to create, and it offered viable results because of how skewed the data is to begin with. It is important to keep in mind that the outputs for predictions are in the range of 0 to 5. Though over half of the dataset had ratings of 4 or higher. As such a flat recommender would also likely perform very well. The most complicated model is the cosine similarity model. This model performed relatively well

for its purpose of finding similar recipes. Though this model could not be run on the full data, and as such may not be suitable as it is not scalable. The model also needs to run over the entire dataset each time to get all cosine similarities making it horribly inefficient.

6 Conclusion

Recommender systems exist in a variety of places. For the space of cooking, the current options available to get a recommendation for what to make are lackluster. They suffer from being clunky and not straight to the point. This makes getting results more difficult for the user. To resolve this issue, we sought to create our own recommendation system and deploy it on a website so that it can be used by anyone.

This was accomplished by taking in a recipe dataset from Kaggle, classifying the cuisines of each recipe, and creating two recommender systems. The first recommender gives the user a list of the most popular recipes. The second is dependent upon the user inputting a list of ingredients they have available and delivers recommendations based on cosine similarity between the input and existing recipes in the dataset. We then deployed this recommender as a website that any person can visit and use.

In the future, we would like to revise and improve this recommender system, both from the perspective of the model and from the perspective of the website. We chose cosine similarity for flexibility, as it could also handle a change in how to vectorize ingredients. We might want to try going for something similar to TF-IDF for a user's history of ingredients. Additionally, we may want to go for a hybrid approach by implementing an account system for users to start building a set of explicit and implicit data for us to use to start mixing in collaborative methods.

References

[1] Li, Shuyang, et al. "Food.com Recipes and Interactions." *Kaggle*, Empirical Methods in Natural Language Processing 2019, 8 Nov. 2019, www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions?select=interactions_test.csv.

[2] "Recipe Ingredients Dataset." *Kaggle*, Yummly, 19 Jan. 2017, www.kaggle.com/kaggle/recipe-ingredients-dataset/home?select=train.json.